what is claimed is:

1. A computer program product for converting between graphical and structural text-based representations of business processes, the computer program product comprising a computer usable medium having computer readable program code means embodied in said medium, and comprising

computer readable program code means for storing and maintaining a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

computer readable program code means for identifying portions of an initial graphical representation as matching features in the set of features,

computer readable program code means for generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation,

computer readable program code means for identifying portions of an initial structural text-based representation of a business process as corresponding to pattern mappings associated with features in the set of features, and

computer readable program code means for generating graphical representations of the identified portions of the initial structural text-based representation by reference to the features associated with the pattern mappings corresponding to the identified portions of the initial structural text-based representation.

2. The computer program product of claim 1, in which the set of identifiable features comprises features selected from: synchronous and asynchronous processes, request/response activities, one-way activities, empty nodes, blocks, iterations, receive events, compensation, correlation, variables, fault handling and transition conditions.

3. The computer program product of claim 2, in which the set of identifiable features and the associated pattern mappings, comprises feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition;

4. The computer program product of claim 2, in which the computer readable program code means for generating structural text-based representations of the identified portions of the initial graph-based representation further comprises means for converting Java code referenced in the initial graphical representation to XPath code in the generated structural text-based representation.

5. The computer program product of claim 4, in which the means for converting Java code to XPath code comprises means for converting Java snippet nodes, and Java assignment and condition expressions.

6. The computer program product of claim 5, in which the initial graphical representation is compatible with the Web Sphere™ Studio Application Developer Integration Edition platform and the generated structural text-based representation is compatible with the Business Process Execution Language for Web Services platform.

7. An import and export tool for exporting from graphical representations of business processes to structural text-based representations and for importing from structural text-based representations of business processes to graphical representations, the import and export tool comprising:

storage and maintenance code for implementing the storage and maintenance of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

graph identification code for identifying portions of an initial graphical representation as matching features in the set of features,

export generation code for generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation,

import identification code for identifying portions of an initial structural text-based representation of a business process as corresponding to pattern mappings associated with features in the set of features, and

import generation code for generating graphical representations of the identified portions of the initial structural text-based representation by reference to the features associated with the pattern mappings corresponding to the identified portions of the initial structural text-based representation.

8. The import and export tool of claim 7, in which the set of identifiable features and the associated pattern mappings, comprises feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an

asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition;

9. The import and export tool of claim 7, in which the export generation code comprises conversion code for converting Java code referenced in the initial graphical representation to XPath code in the generated structural text-based representation.

10. A computer program product for converting from a graphical to a structural text-based representation of business processes, the computer program product comprising a computer usable medium having computer readable program code means embodied in said medium, and comprising

computer readable program code means for storing and maintaining a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

computer readable program code means for identifying portions of an initial graphical representation as matching features in the set of features, and

computer readable program code means for generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation.

11. The computer program product of claim 10, in which the computer readable program code means for generating structural text-based representations of the identified portions of the initial graph-based representation further comprises means for extracting properties from graphical elements in the identified portions of the initial graph-based representation and defining corresponding attributes for elements in the generated structural text based representations.

12. The computer program product of claim 11, in which the set of identifiable features comprises features selected from: synchronous and asynchronous processes, request/response activities, one-way activities, empty nodes, blocks, iterations, receive events, compensation, correlation, variables, fault handling and transition conditions.

13. The computer program product of claim 10, in which the set of identifiable features and the associated pattern mappings, comprises feature and pattern mapping pairs selected from the following set of pairs:

    i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

    ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

    iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

    iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

    v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition;

14. The computer program product of claim 12, in which the computer readable program code means for generating structural text-based representations of the identified portions of the initial graph-based representation further comprises means for converting Java code referenced in the initial graphical

representation to XPath code in the generated structural text-based representation.

15. The computer program product of claim 14, in which the means for converting Java code to XPath code comprises means for converting Java snippet nodes, and Java assignment and condition expressions.

16. The computer program product of claim 15, in which the initial graphical representation is compatible with the Web Sphere™ Studio Application Developer Integration Edition platform and the generated structural text-based representation is compatible with the Business Process Execution Language for Web Services platform.

17. An export tool for exporting from graphical representations of business processes to structural text-based representations, the export tool comprising:

storage and maintenance code for implementing the storage and maintenance of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

graph identification code for identifying portions of an initial graphical representation as matching features in the set of features, and

export generation code for generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation.

18. The export tool of claim 17, in which the set of identifiable features comprises features selected from: synchronous and asynchronous processes, request/response activities, one-way activities, empty nodes, blocks, iterations, receive events, compensation, correlation, variables, fault handling and transition conditions.

19. The export tool of claim 18, in which the set of identifiable features and the associated pattern mappings, comprises feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition;

20. The export tool of claim 17, in which the export generation code comprises conversion code for converting Java code referenced in the initial graphical representation to XPath code in the generated structural text-based representation.

21. An import tool for importing from structural text-based representations of business processes to graphical representations, the import tool comprising:

storage and maintenance code for implementing the storage and maintenance of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

import identification code for identifying portions of an initial structural text-based representation of a business process as corresponding to pattern mappings associated with features in the set of features, and

import generation code for generating graphical representations of the identified portions of the initial structural text-based representation by reference to the features associated with the pattern mappings corresponding to the identified portions of the initial structural text-based representation.

22. The import tool of claim 21, in which the set of identifiable features and the associated pattern mappings, comprises feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition;

23. A computer implemented method for converting from graphical to structural text-based representations of business processes, the method comprising the steps of:

defining and maintaining a data representation of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

identifying portions of an initial graphical representation matching features in the set of features, and

generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation.

24. The method of claim 23, in which the set of identifiable features comprises features selected from: synchronous and asynchronous processes, request/response activities, one-way activities, empty nodes, blocks, iterations, receive events, compensation, correlation, variables, fault handling and transition conditions.

25. The method of claim 24, in which the set of identifiable features and the associated pattern mappings, comprises feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of

the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition;

26. The method of claim 24, in which the step of generating structural text-based representations of the identified portions of the initial graph-based representation further comprises the steps of converting Java code referenced in the initial graphical representation to XPath code in the generated structural text-based representation.

27. The method of claim 26, in which the steps of converting Java code to XPath code comprises the steps of converting Java snippet nodes, and Java assignment and condition expressions.

28. The method of claim 27, in which the initial graphical representation is compatible with the Web Sphere™ Studio Application Developer Integration Edition platform and the generated structural text-based representation is compatible with the Business Process Execution Language for Web Services platform.

29. A computer program product comprising a computer-readable signal-bearing medium, the said medium comprising means for accomplishing the method of claim 23.

30. The computer program product of claim 29 in which the medium is a recordable data storage medium.

31. The computer program product of claim 29 in which the medium is a modulated carrier signal.

32. The computer program product of claim 31 in which the signal is a transmission over a network.

33. The computer program product of claim 32 claim in which network is the Internet.

34. A computer program product comprising a computer-readable signal-bearing transmission over a network, the said product comprising means for accomplishing the method of claim 25.